YouTube Videos Prediction: Will this video be popular?

Yuping Li[1], Kent Eng[1], Liqian Zhang[1]

[1]Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305

Abstract

Being a new type of job, YouTubers earn money through the advertisement and bonus from videos. Hence, the popularity of videos is the top priority for Youtuber. This project attempts to predict the performance of the videos that are going to be uploaded to YouTube. An equation is developed to manually classify all the videos into four classes: non-popular, overwhelming praises, overwhelming bad views, and neutral videos. Title, time gap, category, tags, description, and duration are selected as the features for the algorithm. Several machine learning algorithms are used to predict the performance and backward search is used on features to select the most relevant features. As a result, extreme gradient boosting with three features {time gap, category, description} is chosen due to its optimal balance between cost and performance.

Introduction

As YouTube becomes one of the most popular video-sharing platforms, YouTuber is developed as a new type of career in recent decades. YouTubers earn money through advertising revenue from YouTube videos, sponsorships from companies, merchandise sales, and donations from their fans. In order to maintain a stable income, the popularity of videos become the top priority for YouTubers. Meanwhile, some of our friends are YouTubers or channel owners in other video-sharing platforms. This raises our interest in predicting the performance of the video. If creators can have a preliminary prediction on their videos' performance, they may adjust their video to gain the most attention from the public.

YouTubers concern about how many people watch their videos the most. Therefore, the videos can be grouped into popular and non-popular based on the number of views. Audiences' feedbacks also are important for YouTubers, because the feedbacks reflect the preference of audiences. Therefore, among the popular videos, the videos are further divided into overwhelming praises, overwhelming bad views, and neutral videos based on the feedback.

In order to predict the performances of videos, the videos' properties {title, time gap, category, tags, description, duration} are selected as the inputs of the machine learning algorithm. The multi-classification algorithms {stochastic gradient descent, multilayer perceptron neutron network, decision trees, random forest, extreme gradient boosting, gradient boosting method} are used to output the predicted class {non-popular, overwhelming praises, overwhelming bad views, neutral videos}. In order to optimize the cost of algorithms, feature selection algorithm {backward search} is used to select the most efficient combination of features.

Related work

Predicting popularity of social media contents has attracted wide attention in recently years. The size of gigantic database makes machine learning become a robust tool to deal with the problem.

A. Feature selection and analysis

Appropriate feature selection is an important topic for predicting the social media contents popularity. Chelaru et al. focused on the ranking approaches used SVM, GBRT, Random Forests, etc. and filtered out important social features, such as likes, dislikes, comments.[1] and Flavio et al. explored the importance of UGC (User generated content) as features.[2] Cheng et al. conducted a large-scale analysis of YouTube videos and provided statistics related to videos.[3] Past researches on features delved into the correlation and importance of features and provided the prerequisite information for latter work.

B. Popularity definition and prediction

Popularity has no scientific definition. Researchers tried to define it with different learning algorithms. Peifeng Yin et al. developed Conformer-Maverick model to simulate a voting process in order to rank popularity. But it is based on early voting patterns, without the prior information, the result is not reliable.[4] Yu et al. exploited the social marketing content to judge whether it would be popular, using SVM and Naïve Bayes. Their methods are limited to textual contents. [5] Szabo et al. established a traditional logarithmic model to predict content popularity with YouTube videos.[6] However, studies conducted by Ratkiewicz et al. shown the changes in popularity in fact occurs in bursts, whose magnitude and time-separation are broadly distributed.[7] S. D. Roy et al. developed a transfer model using Twitter trending topics to solve the problem. [8]

Dataset

We used the Trending YouTube Video Statistics [9] dataset from Kaggle and selected the data of tending videos of YouTube from the US and Canada, which includes information of 29089 unique videos. After canceling the video no longer available, 24726 valid samples were remained. Then we divided the dataset to training, valid, and test set with 70%, 20%, and 10% samples of dataset.

We used number of views, likes, dislikes, and comments to evaluate the performance of the videos and classified them into 4 classes. The standard of classification was based on the distribution of training set and real-life experience. Class 0 is defined as non-popular videos whose views are less than 100,000, and the rest of the classes are popular videos. Class 1 is defined as the videos with

overwhelming bad views, whose score is less than 0. This means the videos at least have a similar amount of dislike and like. Class 2 is defined as neutral videos, which cannot be classified as either getting positive or negative responses, such as news which close its response section. Class 3 is defined as the videos with overwhelming praise if the score is above 300, meaning the video has a significant amount of likes or a high ratio of comments' number to views. Either case can represent the video gains lots of praise from viewers. The following function was used for labeling:

$$Score = \frac{Comments}{Views} \times (Likes - 1.5 \times dislikes)$$

$$y = \begin{cases} 0, & Views < 100,000 \\ 1, & Views \geq 100,000 \quad Score < 0 \\ 2, & Views \geq 100,000 \quad 0 \leq Score < 300 \\ 3, & Views \geq 100,000 \quad Score \geq 300 \end{cases}$$

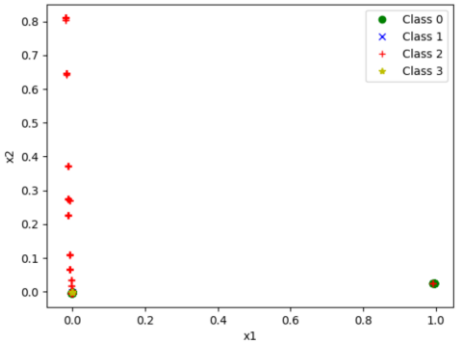
After labeling the dataset, the ratio of each class of the dataset is 21.4%, 1.0%, 67.3%, 10.3% respectively.

Features

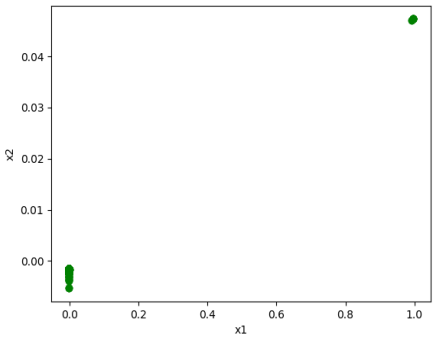
We selected properties of origin videos as features of videos, including video title, tags, description, category, the time gap between trending and publishing date (time gap in days), and duration of videos (in second).

For language features, title, tags, and description, GloVe [10] pretrained dictionary (25 dimension) was used. After embedding each word into a 25-element vector, we add up the all vectors into a 25-element vector for the sentences. Though longer the vector was chosen, better the word was embedded, 25 is a proper length for each language feature for a training set with size of 20,000. No regularization was used as length of each syntagma is also important in this case. Then we combined all features into one vector and get a 79-element vector as the feature for each sample.

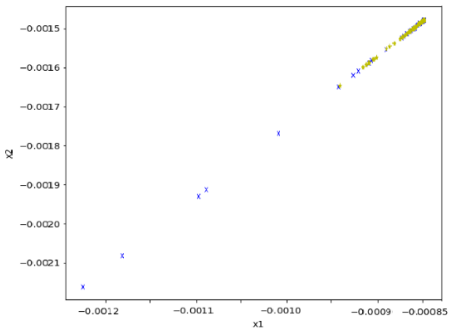
PCA was used to reduce the dimension of features. The first principle component can explain the 97.8% of the variance and the second one can explain 1.9% of the variance of the dataset. Therefore, they can describe the origin data and variance well. The flowing plots shows the discrete state of those 4 classes.



Plot 1-1 distribution of All Classes



Plot 1-2 distribution of Class 2



Plot 1-3 Class 1 and Class 3

According to plot1-1, 1-2, and 1-3, we can see that those 4 classes are hard to separate, especially for class 1 and 3.

Method

Instead of predicting the number of views, likes, dislikes, and comments and then classify the class of the video, we transformed this problem into a multiclass problem. Hence, the following models are all multi-class video.

A. Stochastic Gradient Descent Classifier (SGD)

To begin with, the SGD Classifier was used as the first model. For each sample, theta is updated by the function:

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}, \quad (\text{for every } j)$$

In order to avoid overfitting and overemphasize one specific feature, L_2 penalty was also used, whose constant was set as 7.5.

As the dataset includes three NLP factors, it needs an algorithm which is more tolerance to the outliers with smooth loss. Hence, the modified huber loss was used as the loss function, which is given below.

$$f(p, y) = \begin{cases} 0, & py \geq 1 \\ (1 - py)^2, & py \in [-1, 1) \\ -4py, & py < -1 \end{cases}$$

B. Neuron Network (Multi-layer perceptron classifier, MLPC)

A 3-layer neuron network with logistic function as activated function was used. As there are 6 major features, the first hidden layer had 6 neurons and the second hidden layer had 24 neurons, representing the probability of this feature falling into each class. As we use this function to predict the probability, a logistic function is used as the activation function, which is shown as below:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

C. Decision Tree and Random Forest

Decision tree can be used as a multi-classifier with tree-like structure. Due to the unlimited number of layers, decision tree will be able to achieve high accuracy and cause an overfitting problem. Random forest is an advanced decision tree structure. It randomly selects both samples and features to train different decision trees, and averages the score of different trees, thus reduces overfitting.

D. Gradient Boosting Method and Extreme Gradient Boosting

Boosting trees is a robust method to solve a multi-classification problem. It will train plenty of weak learners and add up the score to achieve the classification goal. It punishes the model with number of entries in nodes and score of different leaves to avoid overfitting. For this problem, we used basic Gradient Boosting Machine and XGBoost model. The loss function of these models is:

$$\text{Cross entropy} = -\sum_{k=1}^{K} y_k \log(p_k(x))$$

Typically, boosting tree method have large loss function flexibility with Taylor expansion in the optimization as below:

$$\text{obj}^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant$$

$$g_i = \partial_{\hat{y_i}^{(t-1)}} \; l(y_i, \hat{y}_i^{(t-1)})$$

$$h_i = \partial^2_{\hat{y_i}^{(t-1)}} \; l(y_i, \hat{y}_i^{(t-1)})$$

To improve efficiency, we only used 80% of the training set to train extreme gradient boosting. The parameters are tuned using grid search and adopted the best combination.

E. Model Improvement: Class weight and Multi-level Binary Framework

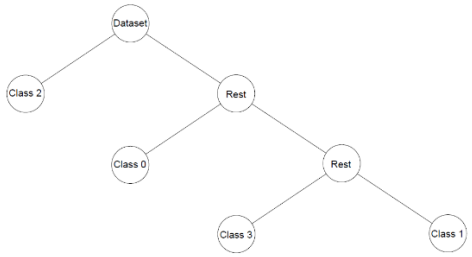
Class Weight:

In the dataset, the ratio of each class of the training set is 21.4%, 1.0%, 67.3%, 10.3%. The algorithm would always tend to predict class 2 and class0 instead of class 1 and 3. This issue of imbalanced class can be solved through putting a class weight into the algorithm. By setting high class weight for class 1 and 3, the loss of predicting wrong class 1 and class 3 would increase.

The simplest method of putting class weight is to take the reciprocal of the ration of each class. And then the class weight can be adjusted by the result of the prediction.

Multi-level Binary Framework:

In the dataset, the ratio of each class of the training set is 21.4%, 1.0%, 67.3%, 10.3%. In order to ensure the algorithms were at least able to classify the largest class in the dataset, a three level of binary framework was constructed. At each level, the class that had the largest occupation would be selected, and the data would be relabeled to a binary classes condition {largest class, rest}. The algorithm {stochastic gradient descent, multilayer perceptron neutron network, random forest, extreme gradient boosting, gradient boosting method} needed to output the predicted class {largest class, rest} in this case. The data that was labeled as {rest} would be passed to next level, and a different class would be selected in the next level. After three level, all four classes would be classified. Since the loss function of the algorithm can also apply on the binary condition, the parameters of algorithm were as same as the one in multi-classification, except the class weight was not applied. The class order of classifying is {2,0,3,1}, and the proportion of class 1 in each layer of the training set is 67.3%, 65.4%, 91.2% respectively. The diagram of the framework is shown as below:

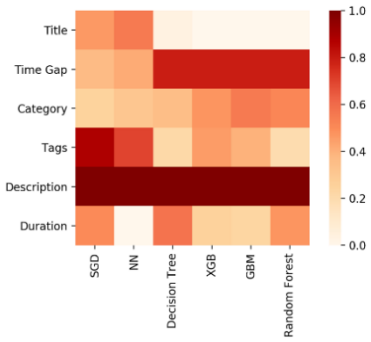


Backward Search on Features

Features was firstly selected by subjective assumption, but it may not be necessary for the algorithm. Therefore, a backward search was used. At each layer, we removed one feature at a time and calculated the f1 score with the remained. Then we took the combination of remained features (one feature removed) that has the highest f1 score as the input of next layer. For example, if there are 3 features {a, b, c} in the input, the algorithm would calculate the f1 score when only using {a, b}, {b, c}, and {a, c}, and select {a, b} as the input if it has the highest f1 score among all situations. The amount of time for a specific feature occurring after each layer would be count as the importance of that feature. In addition, this algorithm was run multiple times with different random state to delimit the occasionally.

Result

In order to visualize the result from backward search on features, heat map is used. After we get the count of each feature occurring at backward search in each model, we regularized each number by max regularization, which means dividing each number by the largest number. Therefore, we would transfer the number in the range of [0, 1]. The result of backward search on features is shown as following:



In order to compare the performance of different model, F1 score was used as a comparison tool in this project. True positive means the model correctly predict the sample is belonging to the specific class. True negative means the model correctly predict the sample that is not belong to that specific class. The false positive and negative is the similar concept, but represent the model have an incorrect predict on the sample. The equation of prevision and recall is shown as following:

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

F1 score can be treated as the weighted average of the precision and recall. The model performs better if the F1 score is closer to 1, and the model perform worse if the F1 score is closer to 0. [11] The equation of F1 score is shown as following:

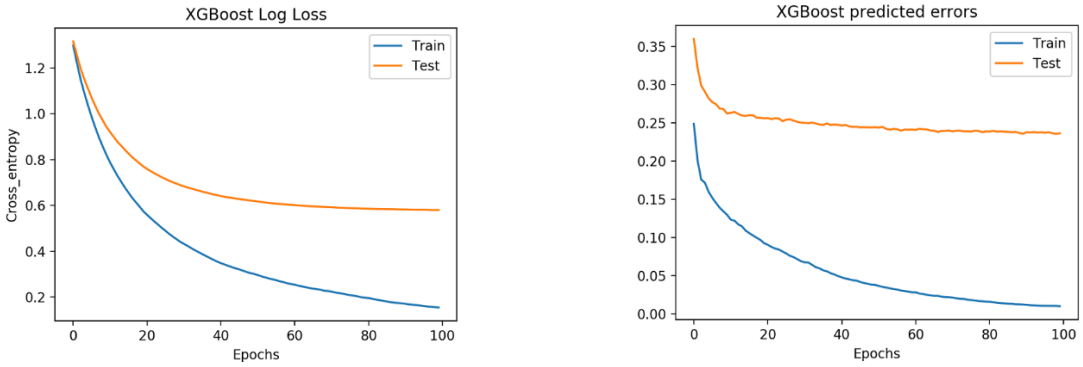
$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

After using several algorithms, the F1 score of each algorithm is shown as following:

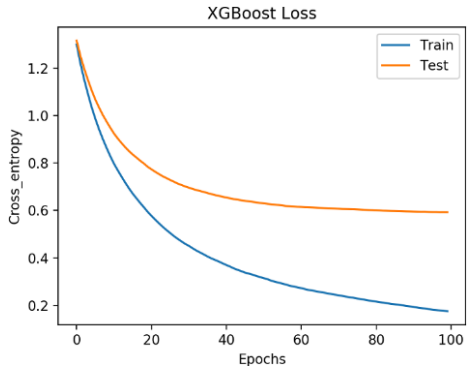
Model	F1 score	F1 score (Binary framework)
Uniform prediction (Base line)	0.303	/
SGDC	0.552	0.557
Neuron Network (MLPC)	0.682	0.679
Decision Tree	0.646	/
Random Forest	0.698	0.723
XGBoost (6 features)	0.741	0.706
XGBoost (3 features)	0.736	/
GBM (6 features)	0.747	0.717
GBM (3 features)	0.727	/

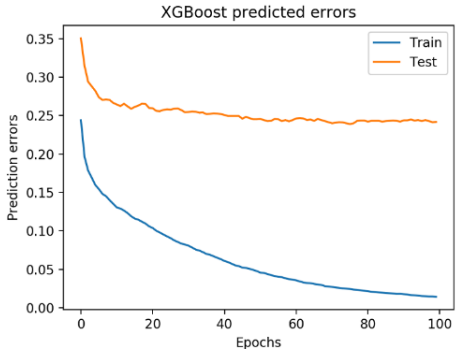
The uniform prediction is assumed the prediction of four classes is uniformly distributed. This prediction and stochastic gradient descent are chosen as base line and base model. Gradient boosting method has the highest F1 score when six features are used, and extreme gradient boosting has the highest F1 score when three features {time gap, category, description} are used.

Finally, extreme boosting algorithm is chosen which will be explained in the later section. A plot of loss value and prediction error is generated for extreme boosting algorithm to check the problem of overfitting, and the plot is shown as following:



After trying to solve the overfitting problem with only consider 3 features, the plot of loss value and prediction error is generated for extreme boosting algorithm with 3 features {time gap, category, description}, and the plot is shown as following:





Discussion

Throughout the prediction process, the six models shown low capability of predicting class 1 with nearly 0 prediction of class 1, which is caused by the imbalanced data. Adding class weight and using multi-layer binary framework are tried to address this problem. After this modification, the models had about 20 predictions of class 1 with about 33% precision. According to the f1 scores, assigning class weight can better improved the performance than binary framework except for Random Forest. It is possibly because the binary framework cannot balance the class as well as class weight do. There is still slight imbalance in each layer, especially in the last layer. For Random Forest, it is essentially binary decision tree, therefore the class weight has less impact on the result.

To solve the overfitting issue, we canceled the features through backward search and used different models as substitutions. We extracted 3 most important features, description, time gap and category, as the major features. According to the plot, we found that in all of the boost models, title has lowest importance while description has the highest importance among all features. Furthermore, we used random forest to replace the decision tree model and XGBoost in place of GBM. When using XGBoost, 80% percent of training data to train XGBoost and got a relatively good result considering the time it saved. The f1 score of XGBoost trained with 6 features is 0.741, and 0.736 with 3 features. Though the f1 score was decreased within 0.1, the efficiency of the models is hugely improved (only 28 features remained). From all the results of model mentioned above, we decided that only using three features, description, time gap and category to train the XGBoost model with class weight set will give us the most satisfying result. However, after these changes, the problem of overfitting still exists.

Conclusion

This project explores how to use machine learning algorithm to predict the video performance for YouTuber. After using several algorithms, model improvements, and backward search on features, extreme gradient boosting with features {time gap, category, description} and class weight {0:0.9, 1:8, 2:0.7, 3:1.7} performs the best with f1 score of 0.736. The features {category, description} briefly represent the content of video, and audience may have tendency to look for certain category or content of video. For {time gap}, as the time pass on, the number of views, likes, dislikes, and comments can only increase for a video. Therefore, it is reasonable that these features {time gap, category, description} become the most important features.

Future Work

The extreme gradient boosting model has large room for improvement. For the features, additional video property {video (series of image)} can also be used as input. This feature represents the content of video in detail. Certain feature {duration} may also be replaced with other feature {subtitle}, because the subtitle can indirectly represent the length of video while it also represents the content of video in detail. Meanwhile, obtaining more video samples for the minority classes through YouTube API can increase the data size and resolve the problem of data imbalance. Furthermore, the convolutional neutral network can also be used as a model to process the features {video (series of image)} that contain images.

Contributions

Eng and Li cleaned the data and obtained features through bigquery. Zhang and Eng built the label equation. Zhang embedded the language feature and analysis features with PCA. Eng, Li, and Zhang trained the model, wrote the poster and the report together.

Code

https://github.com/No21-lqz/CS229AAA

Reference

[1] Chelaru S.V., Orellana-Rodriguez C., Altingovde I.S. (2012) Can Social Features Help Learning to Rank YouTube Videos?. In: Wang X.S., Cruz I., Delis A., Huang G. (eds) *Web Information Systems Engineering - WISE 2012. WISE 2012. Lecture Notes in Computer Science*, vol 7651. Springer, Berlin, Heidelberg. doi:https://link.springer.com/chapter/10.1007/978-3-642-35063-4_40

[2] Figueiredo, F. (2013). On the Prediction of Popularity of Trends and Hits for User Generated Videos. *WSDM '13 Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, 741–746. doi: https://dl.acm.org/citation.cfm?id=2433489

[3] Cheng, X., Dale, C., & Liu, J. (2008). *Statistics and Social Network of YouTube Videos*. 2008 16th Interntional Workshop on Quality of Service, 229-238. doi: https://ieeexplore.ieee.org/document/4539688

[4] Yu B., Chen M., Kwok L. (2011) Toward Predicting Popularity of Social Marketing Messages. In: Salerno J., Yang S.J., Nau D., Chai SK. (eds) *Social Computing, Behavioral-Cultural Modeling and Prediction*. SBP 2011. Lecture Notes in Computer Science, vol 6589. Springer, Berlin, Heidelberg. doi:https://link.springer.com/chapter/10.1007/978-3-642-19656-0_44

[5] Szabo, G., & Huberman, B. A. (2010). Predicting the popularity of online content. *Communications of the ACM*, 80–88. doi: https://dl.acm.org/citation.cfm?id=1787254

[6] Ratkiewicz, J., Menczer, F., Fortunato, S., Flammini, A., & Vespignani, A. (n.d.). Traffic in Social Media II: Modeling Bursty Popularity. doi: https://ieeexplore.ieee.org/abstract/document/5591270

[7] Roy, S. D., Mei, T., Zeng, W., & Li, S. (2013). Towards Cross-Domain Learning for Social Video Popularity Prediction *IEEE Transactions on Multimedia*, 15, 1255–1267. doi: https://www.microsoft.com/en-us/research/wp-content/uploads/2013/08/Towards-Cross-Domain-Learning-for-Social-Video-Popularity-Prediction.pdf

[8] Yin, P., Luo, P., Wang, M., & Lee, W.-C. (2012). A straw shows which way the wind blows: Ranking potentially popular items from early votes.doi: https://www.researchgate.net/publication/221520191_A_straw_shows_which_way_the_wind_blows_Ranking_potentially_popular_items_from_early_votes

[9] YouTube. (2019). Trending YouTube Video Statistics (Version 115) [Daily statistics for trending YouTube videos]. Retrieved from https://www.kaggle.com/datasnaek/youtube-new

[10] Pennington, J., Socher, R., & Manning D., Christopher. (2014). GloVe: Global Vectors for Word Representation. Retrieved from https://nlp.stanford.edu/projects/glove/

[11] Hug, N., & Jalali, Adrin. (2019). scikit-learn. doi: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Library

[12] Hug, N., & Jalali, Adrin. (2019). scikit-learn. doi: https://scikit-learn.org/stable/index.html

[13] McKinney, Wes., ()2019. Pandas. doi: https://pandas.pydata.org/